Memory Management in Java

Objectives

- Study Stack, Static Heap, Dynamic Heap
- Allocation and Deallocation
- Garbage Collection

Memory Management in Java

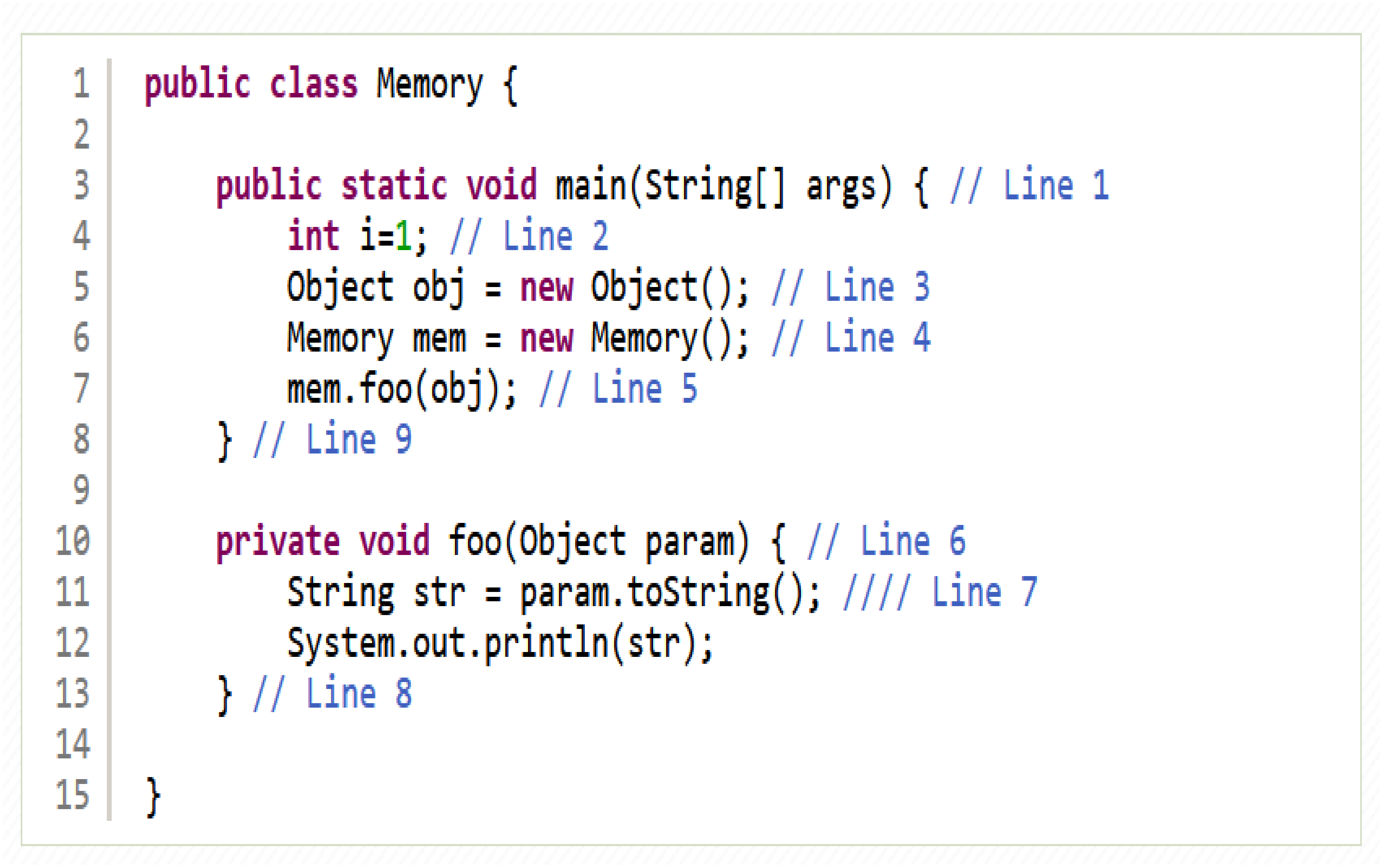
- **Review:** In C, 4 basic regions: Data segment (for global data), code segment (for statements), stack (for local data of functions when they are called), heap (for dynamic data). C/C++ programmers must explicitly manage the heap of a program.
- **How Java heap is managed? (Refer to: http://docs.oracle.com/javase/specs/)**
  - JVM supports the **garbage collector** in order to free Java programmers from explicitly managing heap
  - Java heap is managed by 2 lists: Free block list, Allocated block list
  - Initial, free block list is all the heap
  - After very much times for allocating and de-allocating memory, fragmented and free blocks are not contiguous

Memory Management in Java

- How are data allocated in heap?
  - Way: First fit
  - If there is no blank block is fit, Java memory manager must compact memory in order to create more larger free block
- Heap structure in Java

- Static heap contains class declarations → Invariable, garbage collection is not needed
- Dynamic heap is divided into two sections: The first contains objects and the second contains relations between object and appropriate method in static heap. When an object is not used (garbage), it's memory can be de-allocated.
- When an object is created, a field for reference to the class declaration is automatically added
- The next slide will depict it..

Memory Management in Java



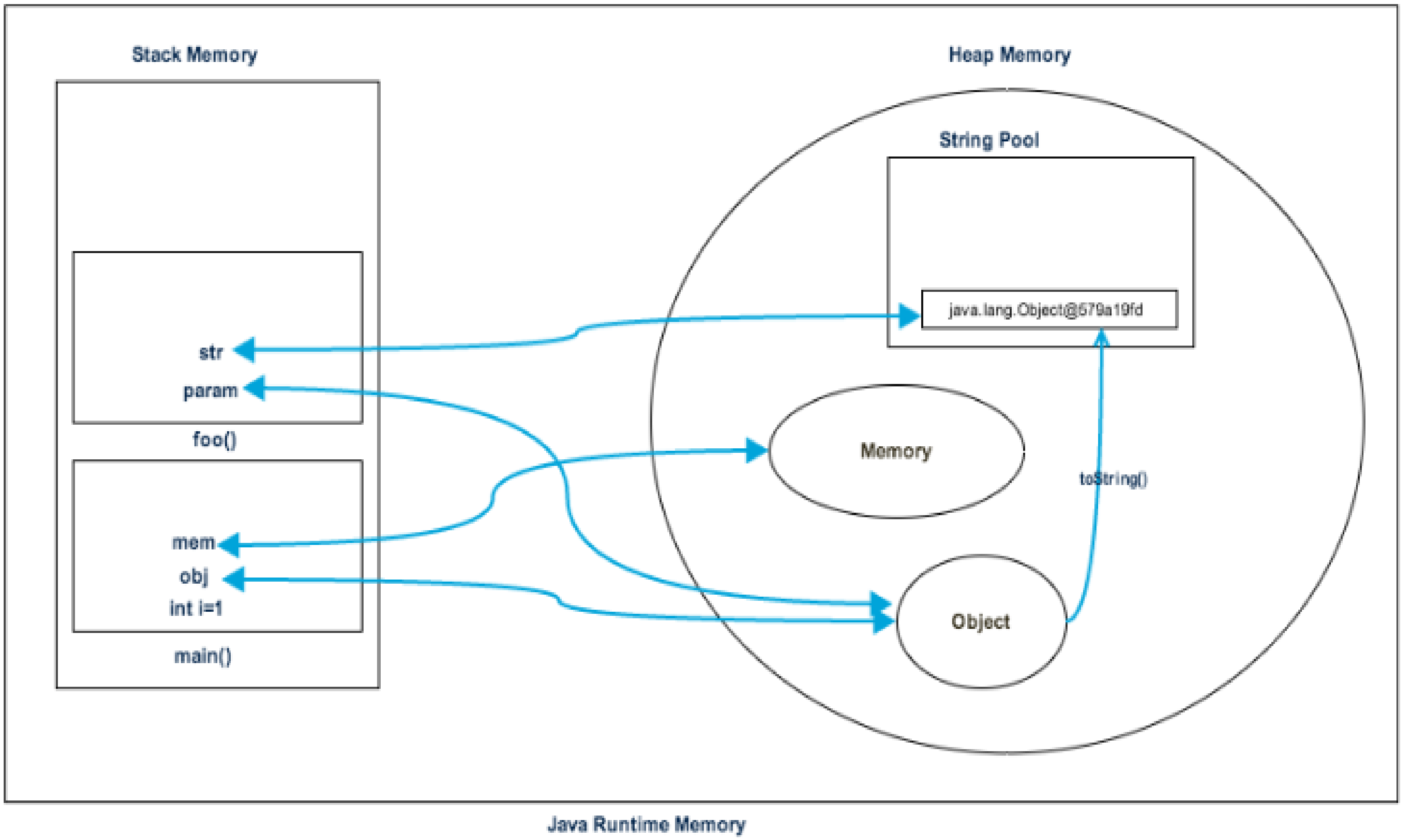
```java
public class Memory {

    public static void main(String[] args) { // Line 1
        int i=1; // Line 2
        Object obj = new Object(); // Line 3
        Memory mem = new Memory(); // Line 4
        mem.foo(obj); // Line 5
    } // Line 9

    private void foo(Object param) { // Line 6
        String str = param.toString(); //// Line 7
        System.out.println(str);
    } // Line 8

}
```

Memory Management in Java



Garbage Collection

- Most modern languages permit you to allocate data storage during a program run. In Java, this is done directly when you create an object with the new operation and indirectly when you call a method that has local variables or arguments.
- Local data of a method include: return data, parameters, variables are declared in the body of the method.
- Local methods are allocated space on the stack and are discarded when the method exits, but objects are allocated space on the heap and have a longer lifetime.

Garbage Collection

- In Java, you <u>never explicitly free the memory</u> that are allocated; instead, Java provides <u>automatic garbage collection</u>.
- The runtime system keeps track of the memory that is allocated and is able to determine whether that memory is still useable.
- Garbage collector has the lowest priority. It runs only when the system heap becomes exhausted.
- A data is treated as garbage when it is out of it’s scope or an object is assigned to null.

Garbage Collection

```
Object obj1 = new Object();
int x= 5;
if (x<10) {
    Object obj2= new Object();
    int y=3;

    .........
}
int t=7;
obj1 = null;
t*=8;
......
```

Scope of a variable begins at the line where it is declared and ends at the closing bracket of the block containing it

obj2, y are out of scope (they are no longer used)

obj1= null → Memory allocated to obj1 is no longer used

Garbage Collection

When does garbage collector execute?

- Garbage collector has the lowest priority. So, it runs only when program's memory is exhausted.
- It is called by JVM only. We can not activate it.